Tanmay Jagtap
TE-2

CNL Assignment-2

Q.1. Explain CRC technique with suitable example.

→ Cyclic Redundancy Check (CRC):

- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to end of data unit. So that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data units divided by the same number. If at this step there's no remainder, the data unit is assumed to be correct & is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit & ∴ must be rejected.

eg: Data to be transmitted = 1010000

Generator polynomial = x^3+1

$= x^3 + 0x^2 + 0x^1 + 1x^0$

CRC generator = 1001 (4-bits)

> If generator if on n-bits, we append n-1 bits

> Sender calculation:

```
1001 | 1010000 000
     ^ 1001
     ------
      001100
       ^1001
       -----
        01010
        ^ 1001
        ------
         0011000
         ^ 1001
         ------
          01010
          ^ 1001
          ------
```

remainder = 011

∴ message to be transmitted:

```
1010000000
    +   011
----------
1010000011
```

⇒ Receiver end:

```
1001 ) 1010000011
     ^ 1001
       ----
       0110000011
     ^  1001
        ----
        0101000011
      ^  1001
         ----
         0011011
       ^   1001
           ----
           1001
         ^ 1001
           ----
           0000   ⟶ No remainder indicates no error.
```

Q.2. What's hamming code? Explain how to correct error bit in hamming code.

⟶ Hamming code: Its a set of error-correction & codes that can be used to detect & correct the errors that might occur during data transmission.

> Redundant bits: They're extra binary bits that've generated & added to the information-carrying bits to ensure that no bits were lost during data transfer.

- The number of redundant bits are calculated by the following formula: $2^r \geq m + r + 1$

  where r = redundant bits & m = data bits.

eg: Suppose m=7, no. of redundant bits $= 2^r \geq m+r+1$
$= 2^4 \geq 7+4+1$

Thus redundant bits = 4

Algo:

1) Write bit pos. starting from 1 in binary form
2) All bits pos that are a power of 2 are marked as priority bits (1, 2, 4, 8) etc
3) All other pos are marked as data bits.
4) Each data bit is included in a unique set of parity bits, as determined by its bit pos. in binary form.
   a) Parity bit 1 includes all bit pos whose LSB is 1 (1, 3, 5, 7, etc)
   b) Parity bit 2 includes all bit pos. whose binary rep includes 1 at 2nd bit from LSB (2, 3, 6, 7, etc)
   - and so on for parity bit 4, 8, 16 etc.
5) Then we check odd/even parity and accordingly give parity bit its value.

→ Error correction:

- Suppose we have following data: 10101101110
- We check values of all parity bits & its corresponding bit pos & if the parity value is right we give value 0, else 1
- In given example we have an error on bit pos 6.
  This is determined as follows:
  $R_0 = 0$; $R_1 = 1$; $R_4 = 1$; $R_8 = 0$
  $\therefore R_8 R_4 R_2 R_1 = 0110$ which is binary 6.
  - Values of R_1 R_2 R_4 & R_8 are determined by explaination given in step 2
  $\therefore$ Data without error: 10101001110
  ↓
  (inverted)

Q.3) Explain checksum in detail:

→ 1) In checksum error detection scheme, the data is divided into k-seg each of m-bits.

2) In sender's end, segments are added using 1's complement to get sum. The sum is complemented to get the checksum.

3) The checksum segment is sent along with data segments

4) At receivers end, all received segments are added using 1's compliment to get the sum. The sum is complemented

5) If result is zero, received data is accepted; else discarded

Eg: If

data

10011001 11100010 00100100 10000100

k=4 ; m=8

Sender

10011001
11100010
100000011
1
01111100
00100100
101000000
10000100
100100100
1
00100101 → sum

∴ check sum = 11011010

Receiver:

10011001
11100010
101111011
1
01111100
00100100
101000000
10000100
100100100
1
00100101
11011010
11111111 → sum

∴ checksum = 00000000

∴ Data is accepted